This Page Is Inserted by IFW Operations
and is not a part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

As rescanning documents *will not* correct images, please do not report the images to the Image Problems Mailbox.

Attorney Docket: 1697P

UNITED STATES PATENT APPLICATION

FOR

METHOD AND SYSTEM FOR ENSURING ACCURATE FONT MATCHING IN DOCUMENTS

Inventor(s):

Brian Berson
Trent Brown
Drew Wilson

Sawyer Law Group LLP
2465 E. Bayshore Road, Suite 406
Palo Alto, California 94303

METHOD AND SYSTEM FOR ENSURING ACCURATE FONT MATCHING IN DOCUMENTS

RELATED APPLICATIONS

The present invention is related to co-pending U.S. Application, entitled METHOD AND SYSTEM FOR IDENTIFYING FONTS IN DOCUMENTS, serial no. _____, filed on August 30, 2000, and assigned to the assignee of the present invention.

FIELD OF THE INVENTION

The present invention relates to storage and use of font identification in documents to ensure accurate font matching.

BACKGROUND OF THE INVENTION

Desktop publishing tools have made document production easier and more widespread. In an ideal workflow for desktop publishing, any document opened on any machine in any application would have the correct fonts activated with the document and have error-free printing. Unfortunately, such an ideal workflow does not exist at present due to problems with font identification within documents.

Using the "wrong" font can cause a document to image incorrectly. Thus, the problems with font identification extend to documents provided in a plethora of environments, including HTML (hyper text markup language) page "documents" and text packet "documents" downloaded to set-top boxes, for example. The wrong font is one whose data (metrics, glyphs, encoding, etc.) differ from the original font such that text won't image in the same manner as with the original font. With most applications in the computing environment today, fonts are referenced using the name of the font. For example, in a QuarkXPress or Adobe

Illustrator document, any fonts used within the document are referenced by the document using the font's name as the key, e.g., if a document used Bauhaus Bold as a font, the name Bauhaus Bold will be referenced within the document. Whenever that document is opened by the application, the application will make a request to the Operating System for a font with the referenced name, e.g. Bauhaus Bold.

While the inclusion of the font name does allow for a rudimentary identification of the font used, a problem arises from the fact that the font name is not unique within the font name space. In other words, there may be numerous fonts that go by the same name, e.g., Bauhaus Bold, that, in fact, behave in different ways. While these fonts may actually be fairly close in specification, the differences can be sufficient to cause some drastic problems. For example, if the width tables of the two Bauhaus Bold fonts are different, the horizontal line spacing could be different when the document is imaged with the incorrect Bauhaus Bold font. The end result could be as drastic as paragraphs of text missing their last line. The ramifications of this, and the potential liability, can be, and often is, quite significant.

Embedding the entire font within a document is a possibility that exists in order to avoid problems with incorrect font identification. Unfortunately, fonts are copyrighted and are licensed for use. By embedding the fonts within documents, there is huge potential for copyright infringement and very little protection for the intellectual property that is the font, since the documents may be used in a variety of worksites. Further, font files can be large, and therefore, embedding all the necessary font files within a document can result in very large document files, as well as a lot of redundancy, since many documents share the same fonts.

Accordingly, a need exists for a method and system for uniquely identifying fonts in a meaningful and efficient manner and for avoiding problems that mismatched fonts cause in document exchange. The present invention addresses such a need.

SUMMARY OF THE INVENTION

The present invention provides method and system aspects for uniquely identifying fonts in a document. The unique identification includes calculating a plurality of attributes for a font and saving the plurality of attributes as a font specification.

With the present invention, unique font identification is provided for documents in an efficient manner to successfully avoid problems associated with font mismatching. The use of font specifications in accordance with the present invention is performed without any reliance on a specific operating platform or on copyrighted information of a font. These and other advantages of the aspects of the present invention will be more fully understood in conjunction with the following detailed description and accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 illustrates a block diagram representation of a document production environment.

Figure 2 illustrates a block flow diagram of a method for identifying fonts in documents in accordance with the present invention.

Figure 3 illustrates a block flow diagram for determining attributes from a font file for each font in accordance with the present invention.

Figures 4, 5, and 6 illustrate block flow diagrams for the storage and usage of font specifications in accordance with the present invention.

DETAILED DESCRIPTION OF THE INVENTION

The present invention relates to font identification in documents, including Internet HTML pages, and text packets downloaded to set-top boxes. The following description is presented to enable one of ordinary skill in the art to make and use the invention and is provided in the context of a patent application and its requirements. Thus, the present invention is not intended to be limited to the embodiment shown, but is to be accorded the widest scope consistent with the principles and features described herein.

The aspects of the font identification through font specifications and the storage and usage of font specifications in accordance with the present invention are preferably performed in a suitable document production environment, e.g., a Macintosh production environment. Figure 1 illustrates a basic block diagram of such an environment that includes a computer processing system 10, such as a Macintosh-type computer, operating in conjunction with a storage device 12, e.g., any suitable storage device for storing a database, such as a hard disk storage device. While a single computer system 10 has been illustrated in Figure 1, this is done for illustrative purposes, and it is typical that multiple users and systems may interact with the data stored in the storage device 12 during the production of a document, as is well appreciated by those skilled in the art. In accordance with the present invention, the environment of Figure 1 performs the aspects as described hereinbelow through appropriate programming stored on a computer readable medium.

In accordance with the present invention, font specifications are created to uniquely identify each font used in a document by calculating a plurality of attributes for a font and saving these attributes as a font specification, as described with reference to an overall block diagram of Figure 2. For each font used in the creation of a document, a font file is opened to initiate the font identification (step 20). It should be appreciated that the reference to a font file includes memory resident and network-based (on the Internet) fonts. The font attributes are then determined by reading, deriving, or calculating the necessary data within the font file (step 22), as described in more detail hereinbelow with reference to Figure 3. The determined font attributes form the font specifications that uniquely identify the font used in the document. These font specifications are saved in a database of font specifications to link the font specifications with a specific font file (step 24). It should be appreciated that the term "database" includes an Internet-based distributed database with varying types of clients. The font file is then closed (step 26).

For purposes of this disclosure, font specifications refer to a set of value: data pairs corresponding to the font attributes for a particular font, including include PostScript name, foundry, version, kind, glyph data, horizontal metrics, vertical metrics, encoding, and kerning. Each unique font identification is formed from a single font specification of the value:data pairs. The following provides the description of the value:data pairs for each of the attributes.

PostScript Name: a font's name as defined by the font. This is an ASCII string, generally less than 40 bytes.

Foundry: the name of the font's foundry, as defined by the font. Some fonts include foundry name, while others have unique signatures which can be used to determine the foundry, as is well understood by those skilled in the art.

Version: the version of the font, as defined by the font. Generally an ASCII string of less than 32 characters.

Kind: the font's format and required technology. An example of technology is PostScript or TrueType. An example of format is Type 1 or Type 3 or Multiple Master (for PostScript fonts). The following are examples of possible data for the kind attribute: PostScript Type 1, PostScript Type 3, PostScript Multiple Master, TrueType, PostScript Type 1 Outline, PostScript Type 3 Outline, bitmap, OpenType PostScript, OpenType TrueType.

Glyph data: a checksum value representing the font's glyphs. Glyphs are the images stored within a font.

Horizontal metrics: a value representing a checksum of the horizontal metrics data.

Vertical metrics: a value representing a checksum of the vertical metrics data.

Encoding: a value representing a checksum of the encoding vector data.

Kerning: a value representing a checksum of the kerning data.

Determination of these attributes (step 22, Fig. 2) occurs as represented by the block flow diagram of Figure 3. As shown, the process includes determining the format of the font file (e.g. PostScript, TrueType, etc.) (step 30). The foundry of the font file is then derived by checking a number of different attributes of the font file (step 31). For example, on a Macintosh, a first check of the "Creator" value of the font file is made and used to identify

the foundry of the font. If this value is not available or does not produce a unique or identifiable foundry, the foundry is derived by using the copyright information stored with the font, as is well understood by those skilled in the art. Next, the PostScript name of the font is read from the appropriate table in the font file (step 32). The version string of the font is also read from the appropriate table in the font file (step 33). A plurality of checksum values are then calculated, including the glyph data checksum from the glyph data in the font file (step 34), the kerning data checksum from the kerning data in the font file (step 35), the horizontal metric data checksum from the horizontal metric data in the font file (step 36), the vertical metric data checksum from the vertical metric data in the font file (step 37), and the encoding data checksum from the encoding data in the font file (step 38).

The checksums are used in the font specifications as a means to identify a font characteristic which affects a font's behavior (glyphs, horizontal and vertical metrics, kerning, encoding.) The use of the checksums provides efficiency, reliability, and safety, since the checksums are very compact representations of a large set of data and can be calculated quickly, the values generated are very likely to be unique for different sets of data, even if that data is very similar, and there is no way to recover any useful portion of the original data, thus avoiding liability over license issues. While there are many different checksum algorithms, each with their own strengths, the following algorithm is suitable for the present invention.

Initialization

1. set checksum to 0.

2. get first 32-bit chunk of data from source.

Loop

1. add next 32-bit chunk of data to checksum value.

2. rotate checksum value to the left by 1 bit. For example, a value of 3 in binary is 0011; rotated to the left by 1 bit is 0100, or 4 in decimal. A binary value of 1000 0000 0000 0000 0000 0000 0000 0001; or 2147483649 in decimal, becomes 0000 0000 0000 0000 0000 0000 0000 0011, or 3 in decimal after rotating.

In addition to the creation of the font specifications to uniquely identify fonts in a document, the storage and usage of the font specifications in order to avoid font mismatching in accordance with the present invention are described in further detail with reference to Figures 4, 5, and 6. The saving of the font specifications created for a document (step 24, Fig. 2) results in the formation of a database (step 40, Fig. 4). Usage of the font specifications occurs from access of the database in order to store font specification references in documents (step 42) and to retrieve a font from font specifications present in a document (step 44), as described in more detail with reference to Figures 5 and 6.

Referring now to Figure 5, when an application saves a document, a list of the fonts used within that document is obtained (step 50, Fig. 5) , for example, by a font architecture or in another example by reading the font name within a native graphic file format. If the font specifications for these fonts have already been calculated and saved in the database (as determined via step 52), the font specifications are retrieved for the fonts from the database (step 54). If the font specifications for these fonts have not already been calculated and

saved in the database, the font specifications for the fonts are calculated (i.e., as described with reference to Fig. 2). Once the font specifications are retrieved or calculated, these font specifications are written into the document (step 56) and the document is saved (step 58). The location and form of the font specifications within a particular document are specific to the application saving the document but generally are similar in that they provide a list of value: data pairs, each pair representing an identifying attribute.

In addition to the use of the font specifications and database when saving a document, the database is also accessed when opening a document. Referring to Figure 6, when an application opens a document, the document is checked for font specifications (step 60, Fig. 6). If the document does contain font specifications, then these font specifications are retrieved from the document at the time of opening the document (step 62). The font specifications are utilized to search and locate the associated fonts from the font specification database (step 64). Once located in the database, the associated fonts are retrieved (step 66) according to normal font retrieval techniques for the application opening the document.

As presented herein, the present invention effectively provides creation of font specifications that uniquely identify fonts used in a document. The present invention further provides efficient storage of the font specifications for unique association with each font in a database. In this manner, use of the proper and desired font within a document is assured.

Although the present invention has been described in accordance with the embodiments shown, one of ordinary skill in the art will readily recognize that there could be variations to the embodiments and those variations would be within the spirit and scope

of the present invention. Accordingly, many modifications may be made by one of ordinary skill in the art without departing from the spirit and scope of the appended claims.